Efficient Evaluation Functions for Evolving Coordination

Adrian Agogino, Kagan Tumer

Abstract— **This paper presents a method for creating evaluation functions that efficiently promote coordination in a multi-agent system, allowing single-agent evolutionary computation techniques to be extended to multi-agent domains. While this problem can be addressed directly by treating the entire multi-agent system as a large single agent, the search space is prohibitively large in most cases. Instead, the proposed method focuses on having each agent use its own evolutionary computation method to maximize its own evaluation function. There are two fundamental issue in this approach: 1) how to create an evaluation function for an agent that is aligned with the global evaluation function and 2) how to create an evaluation function that is sensitive to the fitness changes of the agent, while relatively insensitive to the fitness changes of other agents. If the first issue is not addressed, the evolved agents will not coordinate well. If the second issue is not addressed, the collective evolutionary process will be inefficient and the system will be slow to converge to good solutions. This paper shows how to construct evaluation functions that resolve these issues in dynamic, noisy and communication-limited multi-agent environments. On a rover coordination problem, a control policy evolved using aligned and member-sensitive evaluations outperforms global evaluation methods by up to 400%. More notably, in the presence of a larger number of rovers or rovers with noisy and communication limited sensors, the proposed method outperforms global evaluation by a higher percentage than in noise-free conditions with a small number of rovers.**

Index Terms— **evolution strategies, game theory, multi-agent systems.**

I. Introduction

In many continuous control tasks such as pole balancing, robot navigation and rocket control, using evolutionary computation methods to develop controllers based on neural networks has provided successful results [3], [4], [5]. Extending those successes to distributed domains such as coordinating multiple robots, controlling constellations of satellites, and routing over a data network promises significant application opportunities [6], [7], [8]. The goal in such distributed control tasks is to evolve a large set of agents that collectively strive to maximize a global evaluation function [9], [10], [11]. In this paper we focus on a set of data gathering rovers whose task is to maximize the aggregate information collected by all the rovers.

Approaching the design of a multi-agent system directly by an evolutionary algorithm (e.g., having a population of multi-agent controllers and having the evolutionary operators work directly on the multi-agent controllers to produce a solution with high global fitness) is appealing but impractical at best and impossible at worst. The search space for such an approach is simply too large for all but the simplest problems. A more promising solution is to evolve control policies for individual agents by having each of them use their own fitness evaluation function. The key issue in such an approach is to ensure that the agent fitness evaluation function possesses the following two properties: 1) it is aligned with the global evaluation function, ensuring that the agents maximize their own fitness do not hinder one another and hurt the fitness of the full system; and 2) it is sensitive to the fitness of the agent, ensuring that it provides the right selective pressure on the agent (i.e., it limits the impact of other agents in the fitness evaluation function).

Our domain has a number of properties that make it particularly difficult for evolutionary algorithms:

1) The environment is dynamic, meaning that the conditions under which the agents evolve changes with time. The agents need to evolve general control policies, rather than specific policies tuned to their current environment.
2) The agents' sensors are noisy, meaning that the signals they receive from the environment are not reliable. The agents need to demonstrate that the control policies are not sensitive to such fluctuations in sensor readings
3) The agents have restrictions on their sensing abilities, meaning that the information they have access to is limited. The agents need to formulate policies that satisfy the global evaluation function based on limited, local information.
4) The number of agents in the system can be large. The agents need to decouple the impact of other agents from their fitness functions.

This paper provides methods to evolve control policies in dynamic, noisy environments for large collectives of agents with limited communication capabilities. In Section II we discuss the properties needed for multi-agent evaluation functions and how to evolve agents using evaluation functions possessing such properties along with a discussion of related work. In section III we present the "Rover Problem" where a planetary rovers in a collective use neural networks to determine their movements based on a continuous-valued array of sensor inputs. Section IV presents the performance of the multi-rover system evolved using rover evaluation functions in dynamic, noisy and communication limited domains. The results show the effectiveness of the rovers in gathering information is 400% higher with properly derived rover fitness functions than in rovers using a global evaluation function. Finally Section V we discuss the implication of these results and their applicability to different domains.

II. Evolving in Multi-Agent Systems

Designing control policies for multi-agent systems through evolution can generally be approached in one of the following three ways:

1) One can operate direcly over the entire multi-agent system, treating a single control policy over all agents as an instance of a solution and operate on populations of full-system control policies. In this case, the standard evolutionary algorithms are used to select for the full-system control policy that best satisfies a predetermined global evaluation function.
2) One can operate over agents, treating each agent's control policy as an instance of a solution and operate on populations of single-agent control policies. In this case, the evolutionary algorithms are used to select a control policy based on how a given agent using that control policy satisfies *the predetermined global evaluation function.*
3) One can operate over agents, treating each agent's control policy as an instance of a solution and operate on populations of singe-agents control policies. In this case, the evolutionary algorithms are used to select a control policy based on how a given agent satisfies using that control policy *a specialized agent evaluation function tuned to the fitness of that agent.*

Note that the last two methods are similar in that each agent evolves its own control policy from its own population of policies as shown in Figure 1. These last two methods differ only in the reward being used.

The first method presents a computationally daunting task in all but the simplest problems. Finding good control strategies is difficult enough for single controllers, but the search space become prohibitively large when they are concatenated into an "individual" representing the full multi-agent system. Even if good agents are present in the system, there is no mechanism for isolating and selecting them when the collective to which they belong performs poorly. As a consequence, this approach is practically unworkable in large continuous domains.

The second method addresses part of the issue: Because the agents in the multi-agent system are evolved independently, it avoids the explosion of the state space. However, this method introduces a new problem: How is an agent's evolution guided when the evaluation function depends on the fitness of all the other agents? When there are few agents, this method provides good solutions, but as the number of agents increases, this problem becomes more and more acute. As a consequence, this approach, though preferable to the first approach in some ways, is unlikely to provide good solutions when there are many agents

The third method provides a specialized agent evaluation function for each agent. This approach enables us to create fitness evaluation functions that are more tailored to a specific agent, but introduces a new twist to the problem: How does one ensure that the specialized agent evaluation functions are aligned with the global evaluation function? In other words, the fundamental question is how to guarantee that the evolved multi-agent system using agent evaluation functions will have a high fitness with respect to the global evaluation function. In this paper we discuss the second and third approaches, focusing on how to select agent evaluation function in a formal manner as discussed below.

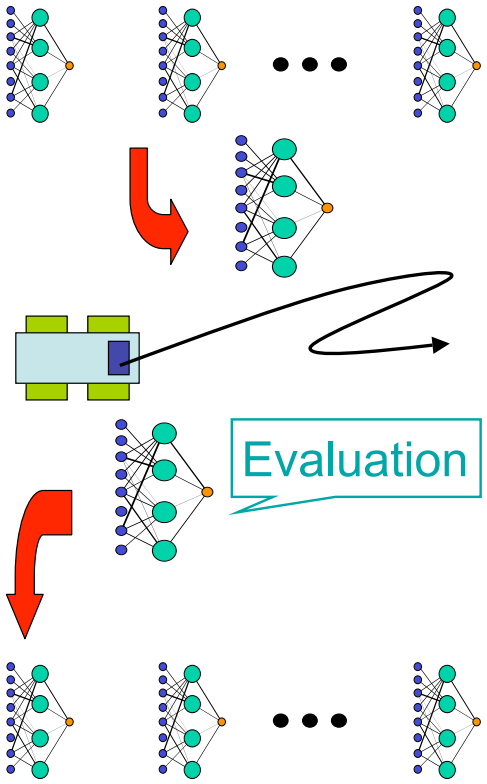

Fig. 1. **Evolution Process for Single Agent.** An agent chooses a control policy from its own population of control policies. It then uses it for control. After evaluating the control policy's effectiveness, the agent updates its population.

A. Agent Evaluation Function Properties

Agent-specific evaluations need to relate to the global evaluation function in a particular way for a multi-agent system to evolve properly. Fortunately by looking at two particular properties of this relation, we can create agent-specific evaluations that will lead to high global performance. We will now formally defines these properties.

Let the **global evaluation function** be given by $G(z)$, where z is the state of the full system (e.g., the position of all the agents in the system, along with their relevant internal parameters and the state of the environment). Let the **agent evaluation function** for agent i be given by $g_i(z)$. First we want the private evaluation functions of each agent to have high *factoredness* with respect to G, intuitively meaning that an action taken by an agent that improves its private evaluation function also improves the global evaluation function (i.e. G and g_i are aligned). Formally, the degree of factoredness between g_i and G is given by:

$$\mathcal{F}_{g_i} = \frac{\int_z \int_{z'} u[(g_i(z) - g_i(z'))\ (G(z) - G(z'))] dz' dz}{\int_z \int_{z'} dz' dz} \quad (1)$$

where z' is a state which only differs from z in the state of agent i, and $u[x]$ is the unit step function, equal to 1 when

$x > 0$. Intuitively, a high degree of factoredness between g_i and G means that a agent evolved to maximize g_i will also maximize G.

Second, the agent evaluation function must be more sensitive to changes in that agent's fitness than to changes in the fitness of other agents in the collective. Formally we can quantify the *agent-sensitivity* of evaluation function g_i, at z as:

$$\lambda_{i,g_i}(z) = E_{z'}\left[\frac{\|g_i(z) - g_i(z - z_i + z_i')\|}{\|g_i(z) - g_i(z' - z_i' + z_i)\|}\right] \quad (2)$$

where $E_{z'}[\cdot]$ provides the expected value possible values of z', and $(z - z_i + z_i')$ notation specifies the state vector where the components of agent i have been removed from state z and replaced by the components of agent i from state z'. So at a given state z, the higher the agent-sensitivity, the more $g_i(z)$ depends on changes to the state of agent i, i.e., the better the associated signal-to-noise ratio for i. Intuitively then, higher agent-sensitivity means there is "cleaner" (e.g., less noisy) selective pressure on agent i.

As an example, consider the case where the agent evaluation function of each agent is set to the global evaluation function, meaning that each agent is evaluated based on the fitness of the full collective (e.g., approach 2 discussed in Section II). Such a system will be fully factored by the definition of Equation 1. However, the agent fitness functions will have low agent-sensitivity (the fitness of each agent depends on the fitness of all other agents).

B. Difference Evaluation Functions

Let us now focus on improving the agent-sensitivity of the evaluation functions. To that end, consider **difference** evaluation functions [9], which are of the form:

$$D_i \equiv G(z) - G(z_{-i} + c_i) \quad (3)$$

where z_{-i} contains all the states on which agent i has no effect, and c_i is a fixed vector. In other words, all the components of z that are affected by agent i are replaced with the fixed vector c_i. Such difference evaluation functions are fully factored no matter what the choice of c_i, because the second term does not depend on i's states [9] (e.g., D and G will have the same derivative with respect to z_i). Furthermore, they usually have far better agent-sensitivity than does a global evaluation function, because the second term of D removes some of the effect of other agents (i.e., noise) from i's evaluation function. In many situations it is possible to use a c_i that is equivalent to taking agent i out of the system. Intuitively this causes the second term of the difference evaluation function to evaluate the fitness of the system without i and therefore D evaluates the agent's contribution to the global evaluation.

Though for linear evaluation functions D_i simply cancels out the effect of other agents in computing agent i's evaluation function, its applicability is not restricted to such functions. In fact, it can be applied to any linear or non-linear global utility function. However, its effectiveness is dependent on the domain and the interaction among the agent evaluation functions. At best, it fully cancels the effect of all other agents. At worst, it reduces to the global evaluation function, unable to remove any terms (e.g., when z_{-i} is empty, meaning that agent i effects all states). In most real world applications, it falls somewhere in between, and has been successfully used in many domains including agent coordination, satellite control, data routing, job scheduling and congestion games [6], [12], [9]. Also note that the computation of D_i is a "virtual" operation in that agent i computes the impact of its not being in the system. There is no need to re-evolve the system for each agent to compute its D_i, and computationally it is often easier to compute than the global evaluation function [12]. Indeed in the problem presented in this paper, for agent i, D_i is easier to compute than G is (see details in Section IV).

C. Related Work

Evolutionary computation has a long history of success in singe agent and multi-agent control problems [13], [14], [15], [16], [17], [18]. Advances in evolutionary computation methods in single agent domains tend to result from improvements in search methods. In [14] this is accomplished through fuzzy rules in a helicopter control problem, while in [13] cellular encoding is used to improve performance on pole-balancing control. Similarly [15] addresses planetary rover control by having genetic algorithms search through a space of plans generated from a planning algorithm.

Evolutionary computation has also been applied to multi-agent domains. In [19] and [20] game theory was used with evolutionary computation in the prisoner's dilemma problem for two-agent and N-agents respectively. In [21] evolutionary methods were used in the domain of scheduling constellations of satellites. Additional advances in evolutionary computation for multi-agent control have been accomplished through the use of domain specific fitness functions. Ant colony algorithms [22] solve the coordination problem by utilizing "ant trails" that provide implicit fitness functions resulting in good performance in path-finding domains. In [16], the algorithm takes advantage of a large number of agents to speed up the evolution process in certain domains, but uses greedy fitness functions that are not generally factored. In [17] beliefs about about other agents are update through global and hand-tailored fitness functions. Also outside of evolutionary computation, coordination between a set of mobile robots has been accomplished through the use of hand-tailored rewards designed to prevent greedy behavior [23]. While highly successful in many domains all of these methods differ from the methods used in this paper in that they lack a general framework for efficient evolution in multi-agent systems.

III. Continuous Rover Problem

In this section, we show how evolutionary computation with the difference evaluation function can be used effectively in the Rover Problem[1]. In this problem, a collective of rovers on a two dimensional plane is trying to observe points of interests (POIs). Each POI has a value associated with it and each observation of a POI yields an observation value inversely related to the distance the rover is from the POI. In

[1]This problem was first presented in [6].

this paper the distance metric will be the squared Euclidean norm, bounded by a minimum observation distance, δ_{min}:[2]

$$\delta(x,y) = min\{\|x-y\|^2, \delta^2_{min}\} \; . \tag{4}$$

The global evaluation function is given by:

$$G = \sum_t \sum_j \frac{V_j}{\min_i \delta(L_j, L_{i,t})} \; , \tag{5}$$

where V_j is the value of POI j, L_j is the location of POI j and $L_{i,t}$ is the location of rover i at time t. Intuitively, while any rover can observe any POI, as far as the global evaluation function is concerned, only the closest observation matters[3].

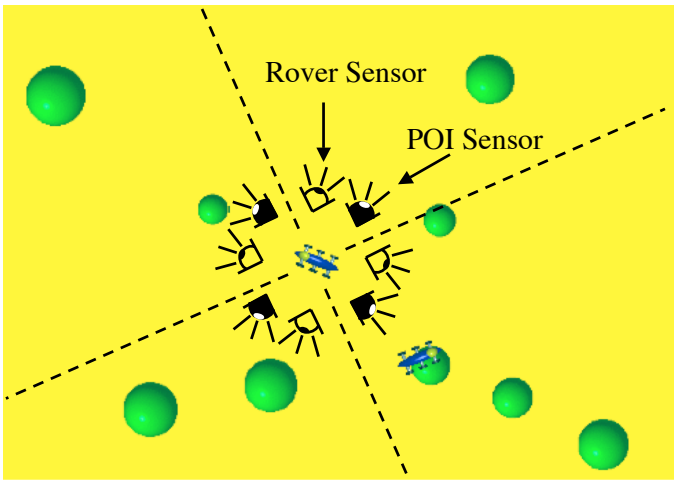

Fig. 2. **Diagram of a Rover's Sensor Inputs.** The world is broken up into four quadrants relative to rover's position. In each quadrant one sensor senses points of interests, while the other sensor senses other rovers.

A. Rover Capabilities

At every time step, the rovers sense the world through eight continuous sensors. From a rover's point of view, the world is divided up into four quadrants relative to the rover's orientation, with two sensors per quadrant (see Figure 2). For each quadrant, the first sensor returns a function of the POIs in the quadrant at time t. Specifically the first sensor for quadrant q returns the sum of the values of the POIs in its quadrant divided by their squared distance to the rover and scaled by the angle between the POI and the center of the quadrant:

$$s_{1,q,j,t} = \sum_{j \in J_q} \frac{V_j}{\delta(L_j, L_{i,t})} \left(1 - \frac{|\theta_{j,q}|}{90}\right) \tag{6}$$

where J_q is the set of observable POIs in quadrant q and $|\theta_{j,q}|$ is the magnitude of the angle between POI j and the center of the quadrant. The second sensor returns the sum of square distances from a rover to all the other rovers in the quadrant at time t scaled by the angle:

$$s_{2,q,i,t} = \sum_{i' \in N_q} \frac{1}{\delta(L_{i'}, L_{i,t})} \left(1 - \frac{|\theta_{i',q}|}{90}\right) \tag{7}$$

where N_q is the set of rovers in quadrant q and $|\theta_{i',q}|$ is the magnitude of the angle between rover i' and the center of the quadrant.

The sensor space is broken down into four regions to facilitate the input-output mapping. There is a trade-off between the granularity of the regions and the dimensionality of the input space. In some domains the tradeoffs may be such that it is preferable to have more or fewer than four sensor regions. Also, even though this paper assumes that there are actually two sensors present in each region at all times, in real problems there may be only two sensors on the rover, and they do a sensor sweep at 90 degree increments at the beginning of every time step.

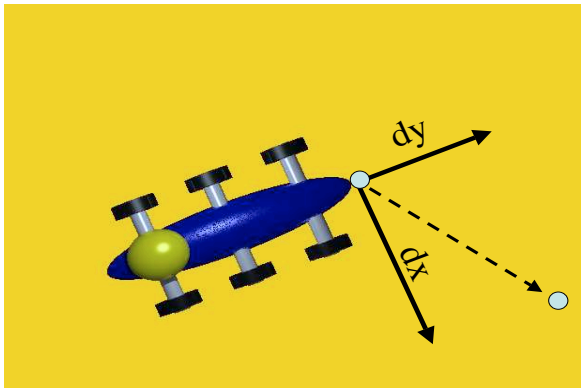

Fig. 3. **Diagram of a Rover's Movement.** At each time step the rover has two continuous outputs (dx, dy) giving the magnitude of the motion in a two directional plane relative to the rover's orientation.

B. Rover Control Strategies

With four quadrants and two sensors per quadrant, there are a total of eight continuous inputs. This eight dimensional sensor vector constitutes the state space for a rover. At each time step the rover uses its state to compute a two dimensional output. This output represents the x, y movement relative to the rover's location and orientation. Figure 3 displays the orientation of a rover's output space.

The mapping from rover state to rover output is done through a Multi Layer Perceptron (MLP), with eight input units, ten hidden units and two output units [4]. The MLP uses a sigmoid activation function, therefore the outputs are limited to the range $(0, 1)$. The actual rover motions dx and dy, are determined by normalizing and scaling the MLP output by the maximum distance the rover can move in one time step. More

[2]The square Euclidean norm is appropriate for many natural phenomenon, such as light and signal attenuation. However any other type of distance metric could also be used as required by the problem domain. The minimum distance is included to prevent singularities when a rover is very close to a POI.

[3]Similar evaluation functions could also be made where there are many different levels of information gain depending on the position of the rover. For example 3-D imaging may utilize different images of the same object, taken by two different rovers.

[4]Note that other forms of continuous reinforcement learners could also be used instead of evolutionary neural networks. However neural networks are ideal for this domain given the continuous inputs and bounded continuous outputs.

precisely, we have:

$$dx = d_{max}(o_1 - 0.5)$$
$$dy = d_{max}(o_2 - 0.5)$$

where d_{max} is the maximum distance the rover can move in one time step, o_1 is the value of the first output unit, and o_2 is the value of the second output unit.

C. Rover Selection

The MLP for a rover is selected using an evolutionary algorithm as highlighted in approaches two and three in Section II. In this case, each rover has a population of MLPs. At each N time steps (N set to 15 in these experiments), the rover uses ϵ-greedy selection ($\epsilon = 0.1$) to determine which MLP it will use (e.g., it it selects the best MLP from its population with 90% probability and a random MLP from its population with 10% probability). The selected MLP is then mutated by adding a value sampled from the Cauchy Distribution (with scale parameter equal to 0.3) to each weight, and is used for those N steps. At the end of those N steps, the MLP's performance is evaluated by the rover's evaluation function and re-inserted into its population of MLPs, at which time, the poorest performing member of the population is deleted. Both the global evaluation for system performance and rover evaluation for MLP selection is computed using an N-step window, meaning that the rovers only receive an evaluation after N steps. The psuedocode for this process is as follows:

```
1. At t=0 initialize N=10 MLPs
2. Pick an MLP using e-greedy alg (e=.1)
3. Randomly modify MLP (mutation)
4. Use MLP to control agent for 15 steps
5. Evaluate MLP performance
6. Re-insert MLP into pool
7. Delete worst MLP from pool
8. Go to step 2
```

While this is not a sophisticated evolutionary algorithm, it is ideal in this work since our purpose is to demonstrate the impact of principled evaluation functions selection on the performance of a collective. Even so, this algorithm has shown to be effective if the evaluation function used by the rovers is factored with G and has high rover-sensitivity. We expect more advanced algorithms from evolutionary computation, used in conjunction with these same evaluation functions, to improve the perform collective further.

D. Evolving Control Strategies in a Collective

The key to success in this approach is to determine the correct rover evaluation functions. In this work we test three different evaluation function for rover selection. The first evaluation function is the global evaluation function (G), which when implemented results in approach two discussed in Section II:

$$G(L) = \sum_t \sum_j \frac{V_j}{\min_i \delta(L_j, L_{i,t})} \tag{8}$$

The second evaluation function is the "perfectly rover-sensitive" evaluation function (P):

$$P_i(L) = \sum_t \sum_j \frac{V_j}{\delta(L_j, L_{i,t})} \tag{9}$$

The P evaluation function is equivalent to the global evaluation function in the single rover problem. In a collective of rover setting, it has infinite rover-sensitivity (in the way rover sensitivity is defined in Section II). This is because the P evaluation function for a rover is not affected by the states of the other rovers, and thus the denominator of Equation 2 is zero. However the P evaluation function is not factored. Intuitively P and G offer opposite benefits, since G is by definition factored, but has poor rover-sensitivity. The final evaluation function is the difference evaluation function. It does not have as high rover-sensitivity as P, but is still factored like G. For the rover problem, the difference evaluation function, D, becomes:

$$\begin{aligned} D_i(L) &= G(L) - G(L - L_i) \\ &= \sum_t \sum_j I_{j,i,t}(z) \left[\frac{V_j}{\delta(L_j, L_{i,t})} - \frac{V_j}{\delta(L_j, L_{k_j,t})} \right], \end{aligned}$$

where k_j is the second closest rover to POI j and $I_{j,i,t}(z)$ is an indicator function, returning one if and only if rover i is the closest rover to POI j at time t. The second term of the D is equal to the value of all the information collected if rover i were not in the system. Note that for all time steps where i is not the closest rover to any POI, the subtraction leaves zero. As mentioned in Section II-B, the difference evaluation in this case is easier to compute as long as rover i knows the position and distance of the closest rover to each POI it can see. In that regard it requires knowledge about the position of fewer rovers than if it were to use the global evaluation function.

In the presence of communication limitations, it is not always possible for a rover to compute its exact D_i, nor is it possible for it to compute G. In such cases, D_i can be compute based on local information with minor modifications, such as limiting the radius of observing other rovers in the system. This has the net effect or reducing the factoredness of the evaluation function while increasing its rover-sensitivity.

IV. Results

We performed extensive simulation to test the effectiveness of the different rover evaluation function under a wide variety of environmental conditions and rover capabilities. The first set of experiments was performed in an episodic environment, which shows the most salient scaling characteristics of the evaluation functions. The second set of experiments was performed in a more realistic non-episodic environment containing sensor noise, controller noise and communication limitations. In all these experiments, each rover had a population of MLPs of size 10. The world was 75 units long and 75 units wide. All of the rovers started the experiment

at the center of the world. Unless otherwise state as in the scaling experiments, there were 30 rovers and 30 POIs in the simulations. The maximum distance the rovers could move in one direction during a time step, d_{max}, was set to 3. The rovers could not move beyond the bounds of the world. The minimum observation distance, δ_{min}, was equal to 5. All results presented were averaged over one hundred trials (except for the seventy rover domains, which were averaged over thirty trials).

A. Episodic Environment

In the first set of experiments the environment was "episodic" (multi-trial) in that at the end of a fixed time interval the position of the rovers and POIs are reset. In most episodic domains, the environment is reset to a fixed starting configuration at the beginning of the episode. However, to make this problem more difficult, we reset the POIs to new random positions at the beginning of each trial. By placing the POIs this way, the rovers have to learn a general policy on how to efficiently navigate using their sensors, and cannot form a specific policy to a single environmental configuration. While episodic domains are less realistic than the non-episodic ones discussed later, they are useful in highlighting the salient properties of the evaluation functions. All results were averaged over at least one hundred independent trials (except for the seventy agent runs where there were thirty trials). For each experiment and trial the weights of the neural network were set to random using the Cauchy distribution (parameter of 0.5).

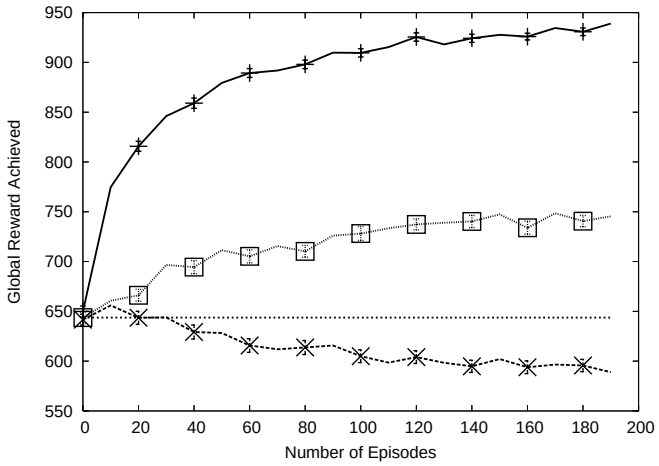

Fig. 4. Performance of a 30-rover collective for all three evaluation functions in episodic environment. Difference evaluation function provides the best collective performance because it is both factored and rover-sensitive.

Figure 4 shows that rovers using D performed best in this scenario. Rovers using D were effective in generalizing the knowledge gained from exploring previous POI configurations and applying that knowledge to new POI configurations. In contrast, rovers using the P rewards were especially ineffective in this scenario. We attribute this to the congested nature of the problem, where the rovers competed rather than cooperating with each other. Since a rover's P rewards only returns the value of what that rover observes, a rover using the P rewards tends to move towards the highest valued POI in its area. However all the other rovers in that vicinity are also moving towards the same high-valued POI, and thus many other POIs are not properly observed.

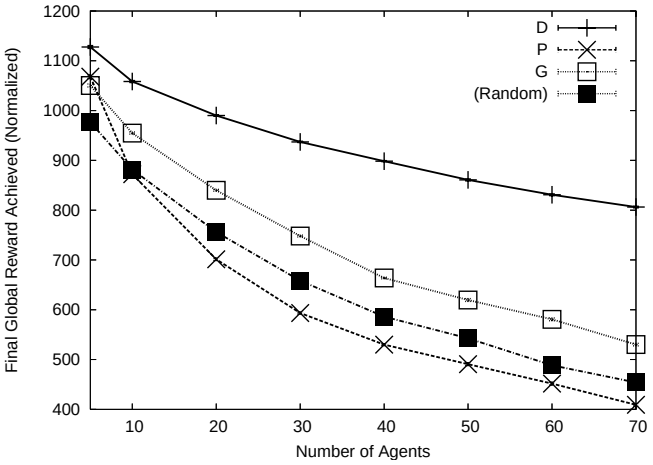

Fig. 5. Scaling properties of the three evaluation functions in episodic environment. The D evaluation function not only outperforms the alternatives, but the margin by which it outperforms them increases as the size of the collective goes up.

Figure 5 shows how varying the number of agents affects the performance of the system for different evaluation functions. The scaling was done between five and seventy agents with the number of POIs being equal to the number of agents. The performance values were then normalized to the thirty agent case to account for the total number of POIs in the system (results were multiplied by $30/n$ for a scale with n agents). The results shows that agents using the D evaluation function performed best for all number of agents. In fact the more agents there were, the more the system benefited from D evaluation as compared to the other evaluation functions. The performance of the systems using G and P evaluations functions quickly deteriorated as the number of agents increased. In the case of the G, this decrease was caused by the learnability of the evaluation function decreasing as the number of agents increased. In large systems it was difficult for an agent to discern its effect on G from the effects of all the other agents. In the case of P the performance decrease was caused by an increase in the amount of competition between agents as the number of agents went up.

B. Non-episodic Environment

In these experiments all evolution occurs within a single trial. The environment and the rover locations are never reset, so rovers have to evolve continuously based on their existing conditions Also the environment was dynamic, meaning that the POI locations and values continuously changed with time. In these experiments there were as many POIs as rovers, and the value of each POI was set to between three and five using a uniformly random distribution. In these experiments, each POI disappeared with probability 2.5%, and another one appeared with the same probability at 15 time step intervals. Because the experiments were run for 3000 time steps, the initial and final environments had little similarities.

Though episodic learning is useful in domains where the simulated environment closely matches the environment in which the rovers will operate, this approach has limited applicability in general. A more desirable approach is for the rovers to evolve efficient policies that are solely based on their sensor inputs and not on the specific configuration of the POIs. The dynamic environment experiments reported here explore this premise and provide rover control policies that can be generalized from one set of POIs to another, regardless of how significantly the environment changes. Figures 6 shows an instance of change in the environment throughout a simulation. The final POI set is not particularly close to the initial POI set and the rovers are forced to focus on the sensor input-output mappings rather than focus on regions in the (x, y) plane.

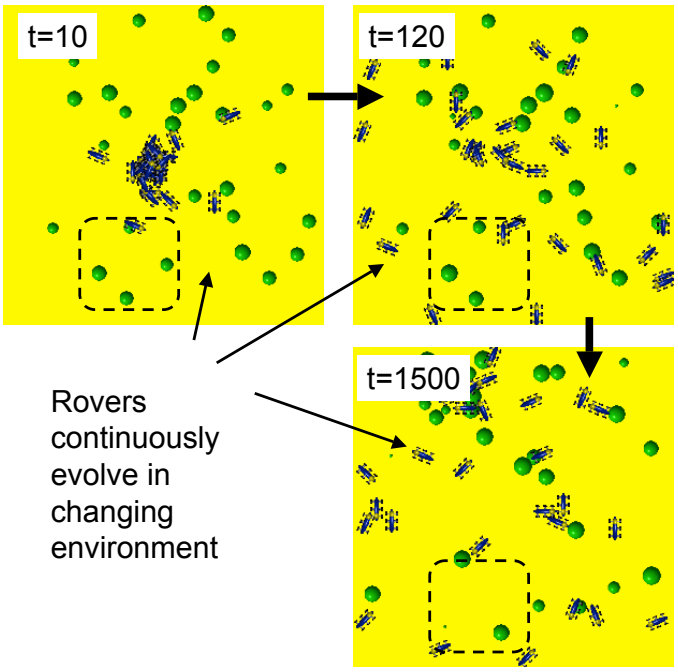

Fig. 6. **Sample POI Placement**. Left-Top: Environment at time = 10. Right-Top: Environment at time = 120. Bottom: Environment at time = 1500. Environment at time step 10 is similar to environment at time step 120, but significantly different than environment at time step 1500. Rovers evolve continuously in single changing environment (no "episodes" or "trials"). Environment and rover locations are never reset. Rovers must to able to use their control policies evolved from earlier time step, in future changed environments.

Note that while it is tempting to compare the relative performance of an evaluation function in episodic domains as compared to in non-episodic domains, this comparison is difficult. In the episodic case, evaluation is done on an episode where the rovers start at a fixed location and move for a fixed number of time steps. In contrast, evaluation in the non-episodic case is done for a window of time where the rovers do not begin the window at their initial locations except at the very beginning of the trial. This difference results in very different learning curves. For example in the episodic case random rovers have a flat performance curve since they start every episode in the same location and perform a similar pattern of actions in every episode. In contrast, the performance of random rovers improves with time in the non-episodic case as they randomly spread out from their initial starting location into the environment.

C. Evolution in Noise Free Environment

The first set of experiments tested the performance of the three evaluation functions in a dynamic noise-free environment for 30 rovers. Figure 7 shows the performance of each evaluation function. In all cases, performance is measured by the same global evaluation function, regardless of the evaluation function used to evolve the system. All three evaluation functions performed adequately in this instance, though D_i outperformed both P and G.

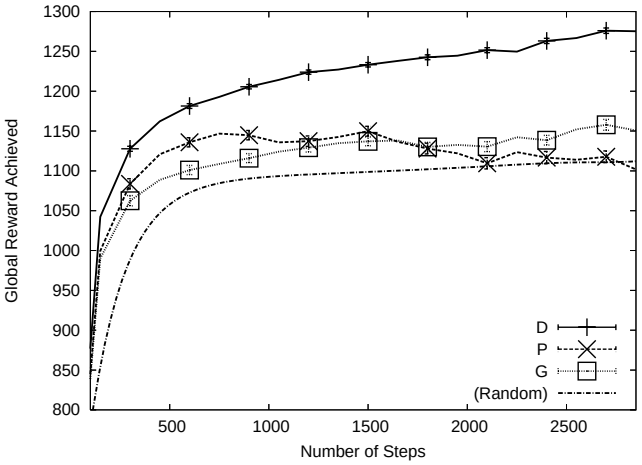

Fig. 7. Performance of a 30-rover collective for all three evaluation functions in noise-free non-episodic environment. Difference evaluation function provides the best collective performance because it is both factored and rover-sensitive.

The evolution of this system also demonstrate the different properties of the rover evaluation functions. After initial improvements, the system with the G evaluation function improves slowly. This is because the G evaluation function has low rover-sensitivity. Because the fitness of each rover depends on the state of all other rovers, the noise in the system overwhelms the evaluation function. P on the other hand has a different problem: After an initial improvement, the performance of systems with this evaluation function decline. This is because though P has high rover-selectivity, it is not fully factored with the global evaluation function. This means that rovers selected to improve P do not necessarily improve G. D on the other hand is both factored and has high rover-sensitivity. As a consequence, it continues to improve well into the simulation as the fitness signal the rovers receive are not swamped by the states of other rovers in the system. This simulation highlights the need for having evaluation function that are both factored with the global evaluation function and have high rover-sensitivity. Having one or the other is not sufficient.

D. Scaling in Noise-free Environments

The second set of experiments focuses on the scaling properties of the three evaluation functions in a dynamic

noise-free environment. Figure 8 shows the performance of each evaluation function at t=3000 for a collective of 10 to 70 rovers. For each different collective size, the results are qualitatively similar to those reported above, except where there are only 5 rovers, in which case P performs as well as G. This is not surprising since with so few rovers, there are almost no interactions among the rovers, and in as large a space as the one used here, the 5 rovers act almost independently.

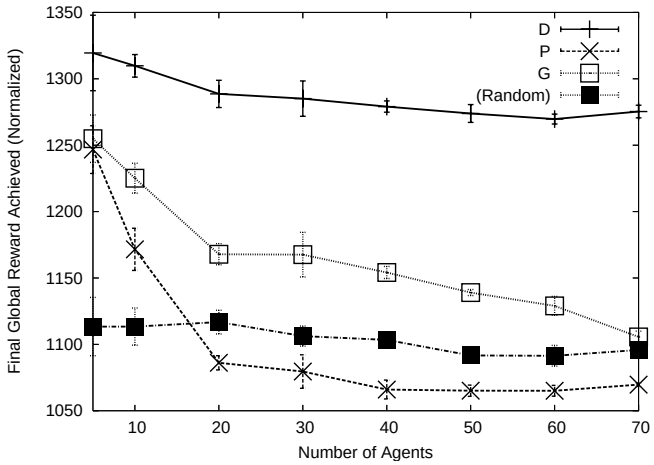

Fig. 8. Scaling properties of the three evaluation functions in non-episodic environment. The D evaluation function not only outperforms the alternatives, but the margin by which it outperforms them increases as the size of the collective goes up.

As the size of the collective increases though, an interesting pattern emerges: The performance of both P and G drop at a faster rate than that of D. Again, this is because G has low rover-sensitivity and thus the problem becomes more pronounced as the number of rovers increases. Similarly, as the number of rovers increases, P becomes less and less factored. D on the other hand handles the increasing number of rovers quite effectively. Because the second term in Equation 3 removes the impact of other rovers from rover i, increasing the number of rovers does very little to limit the effectiveness of this rover evaluation function. This is a powerful results suggesting that D is well suited to evolve large collectives in this and similar domains where the interaction among the rovers prevents both G and P from performing well. This results also supports the intuition expressed in Section II that approach two (i.e., evolving rovers based on the fitness of the full collective) is ill-suited to evolving effective collectives in all but the smallest examples.

E. Evolution in Noisy Environment

The third set of experiments tested the performance of the three evaluation functions in a dynamic environment for 30 rovers with noisy sensors. Figure 9 shows the performance of each evaluation function when both the input sensors and the output values of the rovers have 5% noise added. All three evaluation functions handle the noise well. This result is encouraging in that it shows that not only simple evaluation functions such as P can handle moderate amounts of noise in their sensors and outputs, but so can D. In other words, taking considering the impact of other rovers to yield a factored evaluation function does not cause to compound moderate noise in the system and overwhelm the rover evaluation.

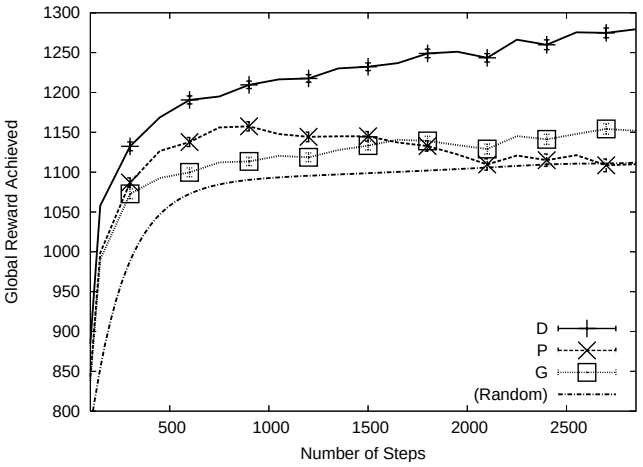

Fig. 9. Performance of a 30-rover collective for all three evaluation functions when the rover sensors and outputs have 5% noise.

Figure 10 shows the noise sensitivity of the three different evaluation functions. The performance is reported as a function of additive noise to sensors as the percentage shown on the x-axis (e.g., 0.5 means the magnitude of the added noise is half that of the sensor value.) The results are shown as the D is the most sensitive to high levels of noise, though even at 80% noise it still far outperforms both G and P. This is an encouraging result in the power of the D evaluation function is that it "cleans up" the evaluation function for a rover (e.g., it has high rover-sensitivity). Adding noise, starts to cancel this property of D, but even when half the signal being noise does not prevent D from far outperforming D and P. Interestingly, rovers using P actually perform marginally better as noise increases, demonstrating the importance of factoredness. Adding noise to the system actually hindered these rovers from learning some counter-productive actions.

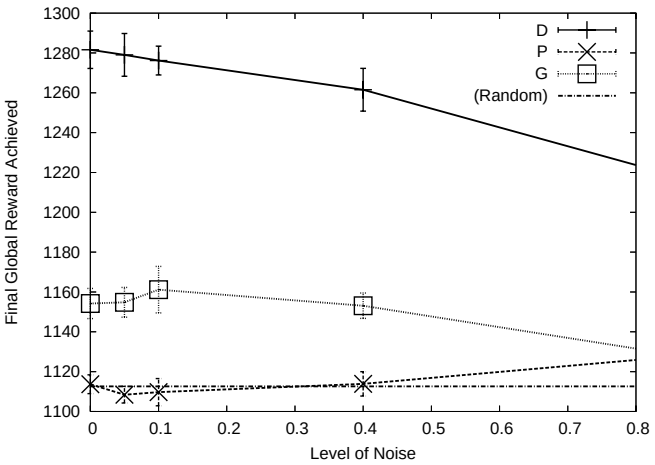

Fig. 10. Sensitivity of the three evaluation functions to the degree of noise in their sensors.

F. Evolution with Communication Limitations

The fourth set of experiments tested the performance of the three evaluation functions in a dynamic environment where not only the rover sensors were noisy, but the rovers were subject to communication limitations. Figure 11 shows the performance of all three evaluation function when the rovers were only aware of other rovers when they were within a radius of 4 units from their current location. This amounts to the rovers being able to communicate with only 1% of the grid. (Because P is not affected by communication restrictions, its performance is the same as that of Figure 7.)

The performance of D is almost identical to that of full communication D. G on the other hand suffers significantly. The most important observation is that communication limited G is no longer factored with respect to the global evaluation function. Though the rover-sensitivity of G goes up in this case, the drop in factoredness is more significant and as a consequence collectives evolved using G cannot handle the limited communication domain.

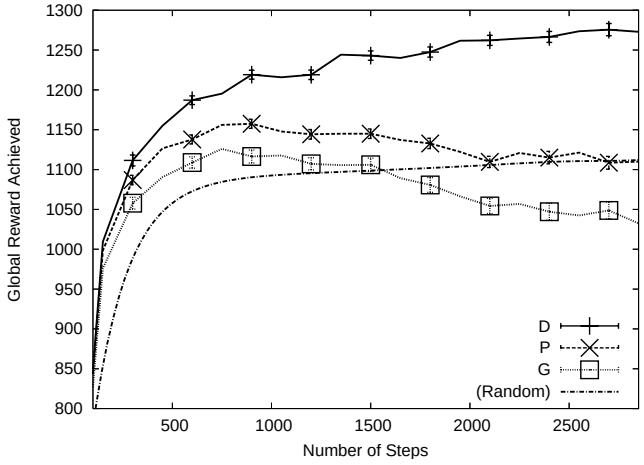

Fig. 11. Results for noisy domain under communication limitations. Rovers can only see other rovers covering an area of 3% of the domain. Difference evaluation is superior since it is both factored and rover-sensitive.

Figure 12 expands on this issue by showing the dependence of all three evaluation function on the communication radius for the rovers (P is flat since rovers using P ignore all other rovers). Using D provides better performance across the board and the performance of D does not degrade until the communication radius is dropped to 2 units. This is a severe restriction that practically cuts the rover from other rovers in the system. G on the other hand needs a rather large communication radius (over 20) to outperform the collectives evolved using P. This results is significant in that it shows that D can be effectively used in many practical information-poor domains where neither G nor "full" D as given in Equation 3 can be accurately computed.

Another interesting phenomenon appears in the results presented in Figure 12, where there is a dip in the performance of the collective when the communication radius is at 10 units for both D and G (the "bowl" is wider for G than D, but it is the same effect). This phenomenon is caused by the interaction between the degree of factoredness of the evaluation functions and their rover-specificity. At the maximum communication radius (no limitations) D is highly factored and has high rover-sensitivity. Reducing the communication radius starts to reduce the factoredness, while increasing the rover-sensitivity. However, the rate at which these two properties change is not identical. At a communication radius of 10, the drop in factoredness has outpaced the gains in rover-sensitivity and the performance of the collective suffers. When the communication radius drops to 5, the increase in rover-sensitivity compensates for the drop in factoredness. This interaction among the rover-sensitivity and factoredness is domain dependent and has also been observed in previous application of collectives [8], [11].

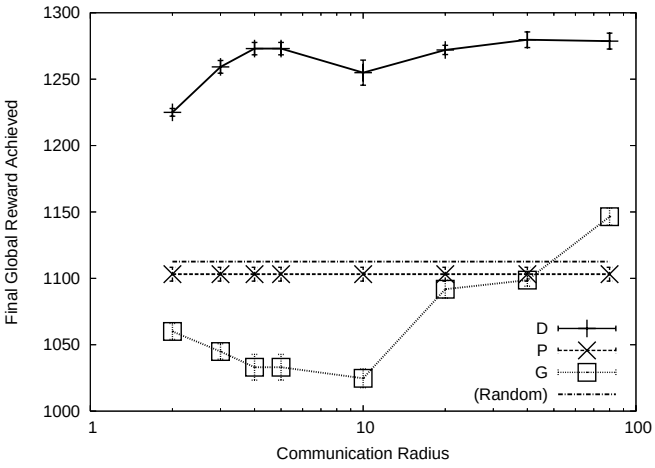

Fig. 12. Sensitivity of the three evaluation functions to the degree of communication limitations. Difference evaluation is not affected by communication limitations by as much as global evaluation.

In addition to measuring the effect of communication limitations between rovers, we also measure the effect on communication limitations on both rovers and POIs, where the evaluation functions only used information within a fixed radius from the rover. The results show in Figure 13 show that this additional communication limitation did not significantly affect the performance of rovers using the D evaluation. However, the decrease in information available actually increase the performance of rovers using both the P and G evaluation functions. This is not surprising in the case of G since the increase in learnability caused by the communication limitation was shown to allow for an increase in performance as previously shown in Figure 12. In contrast the P evaluation already has infinite learnabiliy. Instead decrease in communication caused the evaluation to become more factored. As shown in figure 14, as the communication limitation is reduced the performance of P gets steadily worse.

V. Discussion

Extending the success of evolutionary algorithms in continuous single-controler domains to large, distributed multi-controller domains has been a challenging endeavor. Unfortunately the direct approach of having a population of entire systems and applying the evolutionary algorithm to that population results in a prohibitively large search space

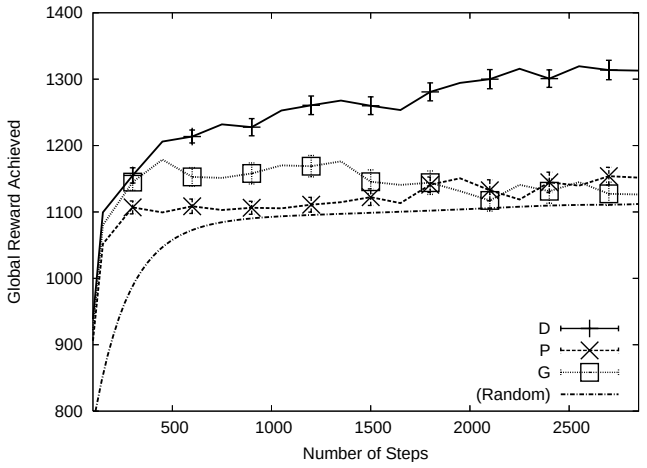

Fig. 13. Results for noisy domain under increased communication limitations. Rovers can only see other rovers or POIs covering an area of 3% of the domain. Difference evaluation is superior since it is both factored and rover-sensitive.

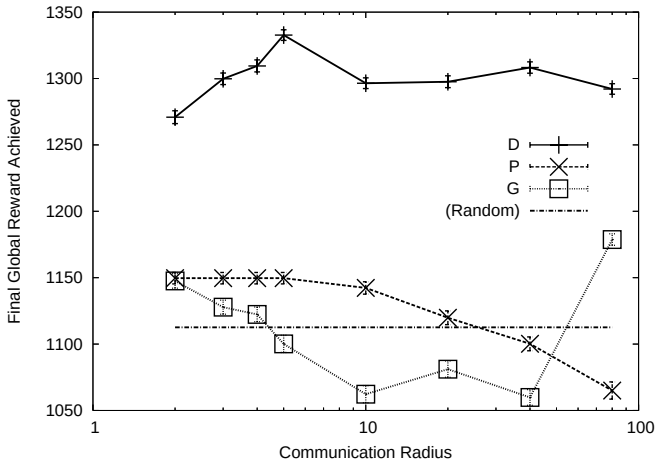

Fig. 14. Sensitivity of the three evaluation functions to the degree of communication limitations. Difference evaluation is not affected by communication limitations by as much as global evaluation.

in most cases. As an alternative, this paper presents a method for providing rover specific evaluation functions to directly evolve individual rovers in system. The fundamental issue in this approach is in determining the rover specific evaluation functions that are both aligned with the global evaluation function and are as sensitive as possible to changes in the fitness of each member.

In dynamic, noise-free environments rovers using the difference evaluation function D, derived from the theory of collectives, were able to achieve high levels of performance because the evaluation function was both factored and highly rover-sensitive. These rovers performed better than rovers using the non-factored perfectly rover-sensitive evaluation and more than 400% better (over random rovers) than rovers using the hard to learn global evaluations.

We then extended these results to rovers with noisy sensors, rovers with limited communication capabilities and larger multi-agent systems. In each instance, the system evolved using D performed better than alternative and in most cases (e.g., larger collectives, communication limited rovers) the gains due to D increase as the conditions worsened. These results show the power of using factored and rover-sensitive fitness evaluation functions, which allow evolutionary computation methods to be successfully applied to large distributed systems in real world applications where communication among the rovers cannot be maintained or where the rover sensors cannot be noise-free.

REFERENCES

[1] G. Baldassarre, S. Nolfi, and D. Parisi, "Evolving mobile robots able to display collective behavior," *Artificial Life*, pp. 9: 255–267, 2003.

[2] T. Balch, "Behavioral diversity as multiagent cooperation," in *Proc. of SPIE '99 Workshop on Multiagent Systems*, Boston, MA, 1999.

[3] K. Stanley and R. Miikkulainen, "Efficient reinforcement learning through evolving neural network topologies," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, San Francisco, CA, 2002.

[4] D. Floreano and F. Mondada, "Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot," in *Proc. of Conf. on Simulation of Adaptive Behavior*, 1994.

[5] F. Gomez and R. Miikkulainen, "Active guidance for a finless rocket through neuroevolution," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Chicago, Illinois, 2003.

[6] A. Agogino and K. Tumer, "Efficient evaluation functions for multi-rover systems," in *The Genetic and Evolutionary Computation Conference*, Seatle, WA, June 2004, pp. 1–12.

[7] A. Martinoli, A. J. Ijspeert, and F. Mondala, "Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots," *Robotics and Autonomous Systems*, vol. 29, pp. 51–63, 1999.

[8] K. Tumer and A. Agogino, "Overcoming communication restrictions in collectives," in *Proceedings of the International Joint Conference on Neural Networks*, Budapest, Hungary, July 2004.

[9] D. H. Wolpert and K. Tumer, "Optimal payoff functions for members of collectives," *Advances in Complex Systems*, vol. 4, no. 2/3, pp. 265–279, 2001.

[10] K. Tumer and D. Wolpert, Eds., *Collectives and the Design of Complex Systems*. New York: Springer, 2004.

[11] K. Tumer and D. Wolpert, "A survey of collectives," in *Collectives and the Design of Complex Systems*. Springer, 2004, pp. 1,42.

[12] K. Tumer and D. H. Wolpert, "Collective intelligence and Braess' paradox," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Austin, TX, 2000, pp. 104–109.

[13] D. Whitley, F. Gruau, and L. Pyeatt, "Cellular encoding applied to neurocontrol," in *International Conference on Genetic Algorithms*, 1995.

[14] F. Hoffmann, T.-J. Koo, and O. Shakernia, "Evolutionary design of a helicopter autopilot," in *Advances in Soft Computing - Engineering Design and Manufacturing, Part 3: Intelligent Control*, 1999, pp. 201–214.

[15] S. Farritor and S. Dubowsky, "Planning methodology for planetary robotic exploration," in *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 124, 2002, pp. 4: 698–701.

[16] A. Agogino, K. Stanley, and R. Miikkulainen, "Online interactive neuro-evolution," *Neural Processing Letters*, vol. 11, pp. 29–38, 2000.

[17] E. Lamma, F. Riguzzi, and L. Pereira, "Belief revision by multi-agent genetic search," in *In Proc. of the 2nd International Workshop on Computational Logic for Multi-Agent Systems*, Paphos, Cyprus, December 2001.

[18] A. Agah and G. A. Bekey, "A genetic algorithm-based controller for decentralized multi-agent robotic systems," in *In Proc. of the IEEE International Conference of Evolutionary Computing*, Nagoya, Japan, 1996.

[19] P. Harrald and D. Fogel, "Evolving continuous behaviors in the iterated prisoner's dilemma," *BioSystems: Special Issue on the Prisoner's Dilemma*, vol. 37, no. 1-2, pp. 135–145, 1996.

[20] X. Yao and P. Darwen, "An experimental study of n-person iterated prisoner's dilemma games," *Informatica*, vol. 18, pp. 435–450, 1994.

[21] A. Globus, J. Crawford, J. Lohn, and A. Pryor, "Scheduling earth observing satellites with evolutionary algorithms," in *Proc. of International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 2003.

[22] M. Dorigo and L. M. Gambardella, "Ant colony systems: A cooperative learning approach to the travelling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.
[23] M. J. Mataric, "Coordination and learning in multi-robot systems," in *IEEE Intelligent Systems*, March 1998, pp. 6–8.